**Liceul Teoretic de Informatică “Grigore Moisil” Iaşi**

**Teză la informatică**

**Clasa Grafuri neorientate**

**Elevi: – XI E, - XI E**

Prof. Tufescu Lăcrămioara

**Cuprins**

|  |  |
| --- | --- |
| **1. Tema proiectului** | **1** |
| **2. Noțiuni fundamentale. Programarea orientată obiect** | **2** |
| **3. Headerul clasei** | **7** |
| **4. Definirea functiilor membru** | **8** |
| **5. Sursa aplicatiei** | **9** |
| **6. Bibliografie** | **10** |
|  |  |

**Tema proiectului:**

**Clasa Garfuri neorientate**

**Programarea orientată obiect**

**Noțiuni teoretice.**

Programarea Orientată Obiect (POO) este o metodă de proiectare și implementare în care programele sunt reprezentate sub forma unor colecții de obiecte care interacționează între ele prin intermediul mesajelor.

Limbaje de programare orientate obiect:

* C++
* C#
* Java

Principalele concepte care stau la baza POO sunt:

* Abstractizarea
* Încapsularea
* Modularitatea
* Ierarhizarea

**Abstractizarea** – Este posibilitatea ca un program să ignore unele aspecte ale informației pe care o manipulează, adică posibilitatea de a se concentra asupra esențialului. Fiecare obiect în sistem are rolul unui “actor” abstract, care poate executa acțiuni, își poate modifica și comunica starea și poate comunica cu alte obiecte din sistem fără a dezvălui cum au fost implementate acele facilitați. Procesele, funcțiile sau metodele pot fi de asemenea abstracte, și în acest caz sunt necesare o varietate de tehnici pentru a extinde abstractizarea.

**Încapsularea** – numită și *ascunderea de informații*: Asigură faptul că obiectele nu pot schimba starea internă a altor obiecte în mod direct (ci doar prin metode puse la dispoziție de obiectul respectiv); doar metodele proprii ale obiectului pot accesa starea acestuia. Fiecare tip de obiect expune o interfață pentru celelalte obiecte care specifică modul cum acele obiecte pot interacționa cu el.

**Modulatitatea** - Modalitate de a grupa abstracţiuni legate logic între ele.

**Ierarhizarea**- Reprezintă o ordonare a abstracţiunilor.

Principalele tipuri sunt:

* Moştenirea (ierarhia de clase) relaţie între clase în care o clasă împărtăşeşte structura şi comportarea definită în una sau mai multe clase (semantic implică o relaţie de tip “is a”).
* Agregarea (ierarhia de obiecte) relaţie între două obiecte în care unul dintre obiecte aparţine celuilalt obiect. (semantic implică o relaţie de tip “part of”).

**Obiecte** - Un obiect este o reprezentare a unei entități din lumea reală asupra căruia se poate întreprinde o acţiune. Un obiect este caracterizat de:

* nume
* atribute (date) - valorile atributelor la un moment dat definesc o stare
* metode (servicii, operatii)

**Clasa** - O clasă este o colecție de obiecte cu aceiași structură (caracteristici) și același comportament (metode sau operatii).

C++ se distinge prin faptul că permite controlul accesului atât la datele member, cât şi la funcţiile membre ale unei clase. În acest scop se folosesc specificatorii de control: **private**, **public**, **protected**.

Efectul acestor specificatori asupra accesului unui membru este:

* public: membrul poate fi accesat de orice funcţie din domeniul de declaraţie a clasei
* private: mebrul este accesibil numai funcţiilor membre şi prietene clase
* protected: membrul este accesibil atât funcţiilor membre şi prietene clasei, cât şi funcţiilor membre şi prietene claselor derivate din clasa respectivă

O funcţie membră a unei clase are acces la toate datele membre a oricărui obiect din clasa respectivă, indiferent de specificatorul de acces.

**Constructor şi destructor**

Pentru crearea, inițializarea, copierea și distrugerea obiectelor, în C++ se folosesc funcții membre speciale, numite constructori și destructori:

* Funcția de tip constructor se apelează automat la crearea fiecărui obiect al clasei, creare care poate fi statică sau dinamică (folosind operatorul new).
* Funcția de tip destructor se apelează automat la eliminarea unui obiect al clasei, la încheierea timpului de viață sau, în cazul variabilelor dinamice, este solicitat prin program (cu operatorul delete).

Aceste tipuri de functii se declara si se definesc asemanator cu celelalte functii membre, dar au cateva particularitati care le diferentiaza de acestea prin:

1. Numele functiilor constructor sau destructor coincide cu numele clasei careia îi apartine; în plus destructorii se diferentiaza prin faptul ca numele lor este precedat de caracterul “ ~ ”.

2. În declaratiile si definitiile constructorilor si destructorilor nu se specifica nici un tip de rezultat, nici macar tipul void.

3. Constructorii si destructorii nu pot fi mosteniti, dar pot fi apelati de clasa derivata.

4. Nu se pot utiliza pointeri catre constructori sau destructori.

5. Constructorii pot avea parametri, inclusiv parametri impliciti, si pot fi supradefiniti. Destructorii nu au aceste proprietati.

6. Un constructor fara parametri se numeste constructor implicit.

7. Daca o clasa nu dispune de constructori si destructori definiti, compilatorul va genera automat un constructor implicit, respectiv un destructor, care sunt si functii publice.

8. De obicei, constructorii si destructorii se declara publici, deoarece, daca s-ar declara privati atunci nu s-ar mai putea declara obiecte de tipul respectiv.

**Supraîncărcarea operatorilor**

Operatorii sunt notaţii concise, infixate, pentru operaţii matematice uzuale. Limbajul C++, ca orice limbaj de programare asigură un set de operatori pentru tipurile primitive. În plus, faţă de limbajul C, C++ oferă posibilitatea asocierii operatorilor existenţi cu tipurile definite de utilizator.

Limbajul C++ introduce următorii operatori noi:

1. new

2. delete - pentru gestiunea memoriei dinamice

3. operatorul de rezoluţie (::)

4. operatorii de acces la membri: .\* şi ->\*

Limbajul C++ permite programatorului să definească diverse operații cu obiecte ale claselor, folosind simbolurile operatorilor standard. Un tip clasă se poate defini împreună cu un set de operatori asociați, obtinuți prin supraîncărcarea (overloading) operatorilor existenți. În acest fel, se efectuează operații specifice cu noul tip la fel de simplu ca în cazul tipurilor standard.

**Functii friend**

Funcțiile friend (prieten) sunt funcţii asociate unor clase care au acces la datele şi metodele protejate ale acelor clase deşi nu sunt funcţii membre ale acelei clase.

Tipuri de funcții prieten:

* funcții globale
* funcții membre ale altor clase

**Bibliografie**

1. [**http://tufesculacramioara.wikispaces.com**](http://tufesculacramioara.wikispaces.com) **[consultat la data de 01-03-2017]**
2. [**www.liis.ro/~infosuport**](http://www.liis.ro/~infosuport) **[consultat la data de 03-04-2017]**
3. [**www.info.mcip.ro**](http://www.info.mcip.ro) **[consultat la data de 06-05-2017]**

**BAREM DE NOTARE: 100 puncte**

10 p – structura clasei

(alegerea adecvată a datelor şi metodelor şi a modului lor de implementare)

20 p – implementarea corectă a funcţiilor membru

10 p – implemetarea meniului

20 p – aplicaţia corect implementată

10 p – funcţionarea corectă a aplicaţiei

10 p – dosarul de prezentare în formatul precizat

10 p – prezentarea/susţinerea adecvată lucrării

10 p – oficiu

**TERMEN DE REALIZARE:** 21 mai 2017

**PRECIZĂRI:**

Creati o arhivă conţinând documentaţia Word şi proiectul aplicaţiei voastre; denumiţi arhiva cu numele vostru şi trimiteţi-o pe mail la adresa **lacratufescu@gmail.com**